

# EzTemplate Documentation

## Introduction

EzTemplate is designed to help users create easily templates that they will be using in vcs.

EzTemplate first sub-package: "Multi" is designed to create template for "multiple" plots per page

EzTemplate are easy to use.

They consist in a 2 steps process

1. Initialization of the object and setting up the values to use
2. Getting the vcs template object

All EzTemplate object have a preview capability

In the examples here we will only use the preview capability, but we will show (commented out) a example line on how to use the templates in vcs

In these examples "s" is the variable to be plotted and "iso" is the graphic method you would like to use to display "s"

## The Multi Object

## Definitions

Multi Object allow the user to display multiple plots on a single page

In Version 1 a few assumption are made:

- The legend is either at the bottom or on the right side of the plot (always centered)

The VCS Canvas has multiple sections or attributes with default values in between parantheses

- row (3) columns (2)
- The **margins**: **top** (0.05), **bottom** (0.075), **left** (0.033) and **right** (0.05)
  - ◆ in % of the page
  - ◆ The Bottom and Right Margins are used as a space to dispaly the global horizontal/vertical legend
- The **spacing**: **horizontal** (0.05) and **vertical** (0.035)
  - ◆ Spacing defined the space between 2 plots on a row or a column
  - ◆ in % of the page
- The **legend**: **direction** ("horizontal"), **thickness** (0.2), **stretch** (0.8) and **fat** (0.05)
  - ◆ Direction of the legend can be horizontal or Vertical
  - ◆ Thickness: in % of the space allocated for it
  - ◆ Stretch: in % of the space allocated for it
  - ◆ Fat: only used for "local" (inside the grid) plot, % of the grid space to use for the legend

- Functions: `get`, `preview`
  - ◆ The `get` function
    - ◊ creates and returns the template object associated with the arguments: **row/column**
    - ◊ if no row/column arguments then creates in order (each call) the template starting from top left, going right then down
    - ◊ **legend** keyword can be passed as "local" indicating that this specific plot needs to have a legend associated with it. this will reduce the area allocated for the data plotting itself, the direction of the legend will be that of the `legend.directive` attribute at the time the function is called
  - ◆ The `preview` function
    - ◊ Generates the templates (if None already generated with a call to `get`) and output a preview result to the file specified by the `out` keyword

## Examples

Basic

In this example we simply plot 3 columns and 4 rows

```
import EzTemplate,vcs

## Initialize VCS
x=vcs.init()

M=EzTemplate.Multi(rows=4,columns=3)

for i in range(12):
    t=M.get()
    ##     x.plot(s[i],t,iso)
M.preview('test')
raw_input('Done')
```



In this example will show how to plot a legend on the side of each plot

```
import EzTemplate,vcs  
  
## Initialize VCS  
x=vcs.init()  
  
M=EzTemplate.Multi(rows=4,columns=3)  
M.legend.direction='vertical'  
for i in range(12):
```

```
t=M.get(legend='local')
##     x.plot(s[i],t,iso)
M.preview('test2')
```

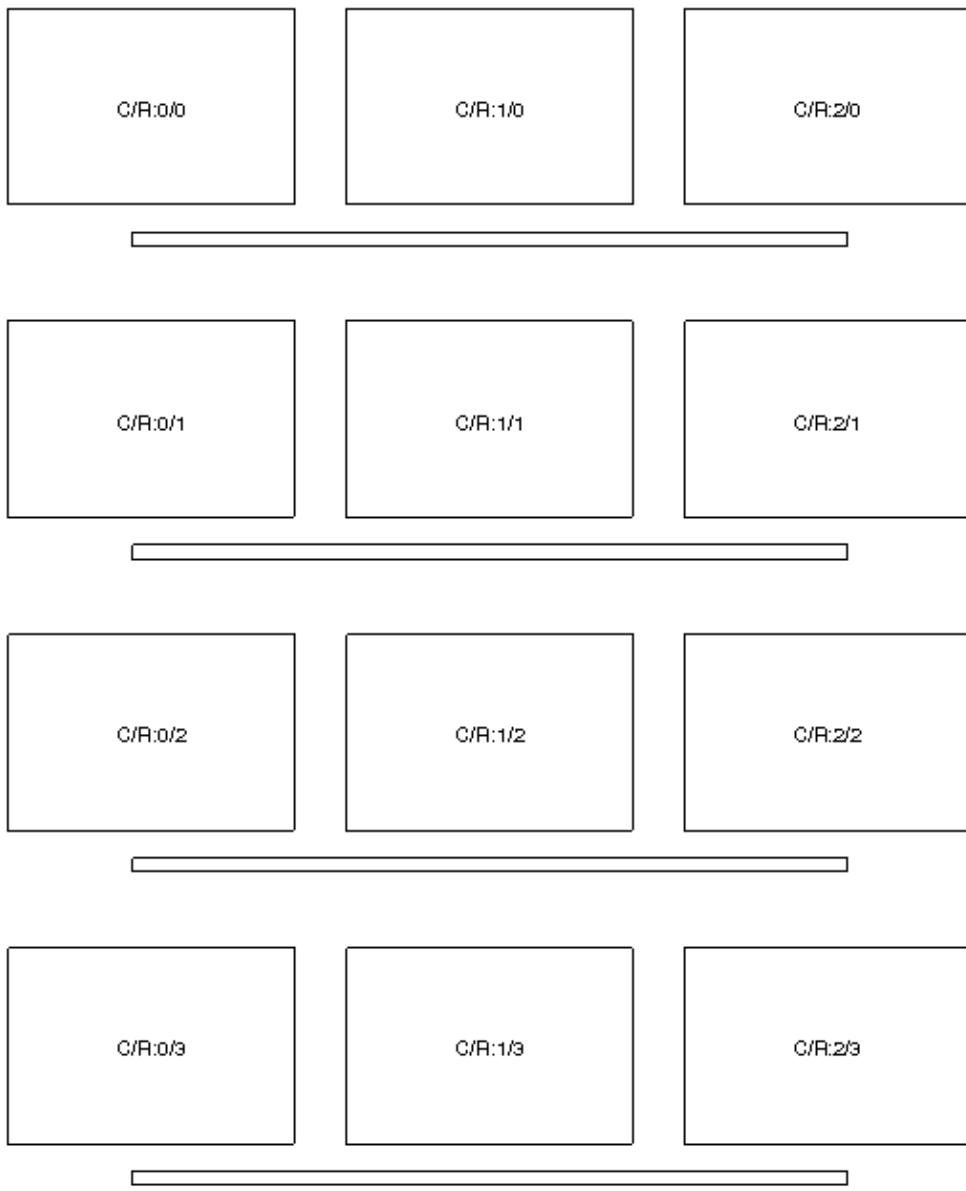


In this example we show how to draw an horizontal legend bar on each row

```
import EzTemplate,vcs
## 12 plot one legend per row
```

```
## Initialize VCS
x=vcs.init()

M=EzTemplate.Multi(rows=4,columns=3)
M.legend.stretch=2.5 # 250% of width (for middle one)
for i in range(12):
    t=M.get(legend='local')
    if i%3 !=1:
        t.legend.priority=0 # Turn off legend
##    x.plot(s[i],t,iso)
M.preview('test3')
raw_input('Done')
```



In this example we show how to draw one legend per row vertically

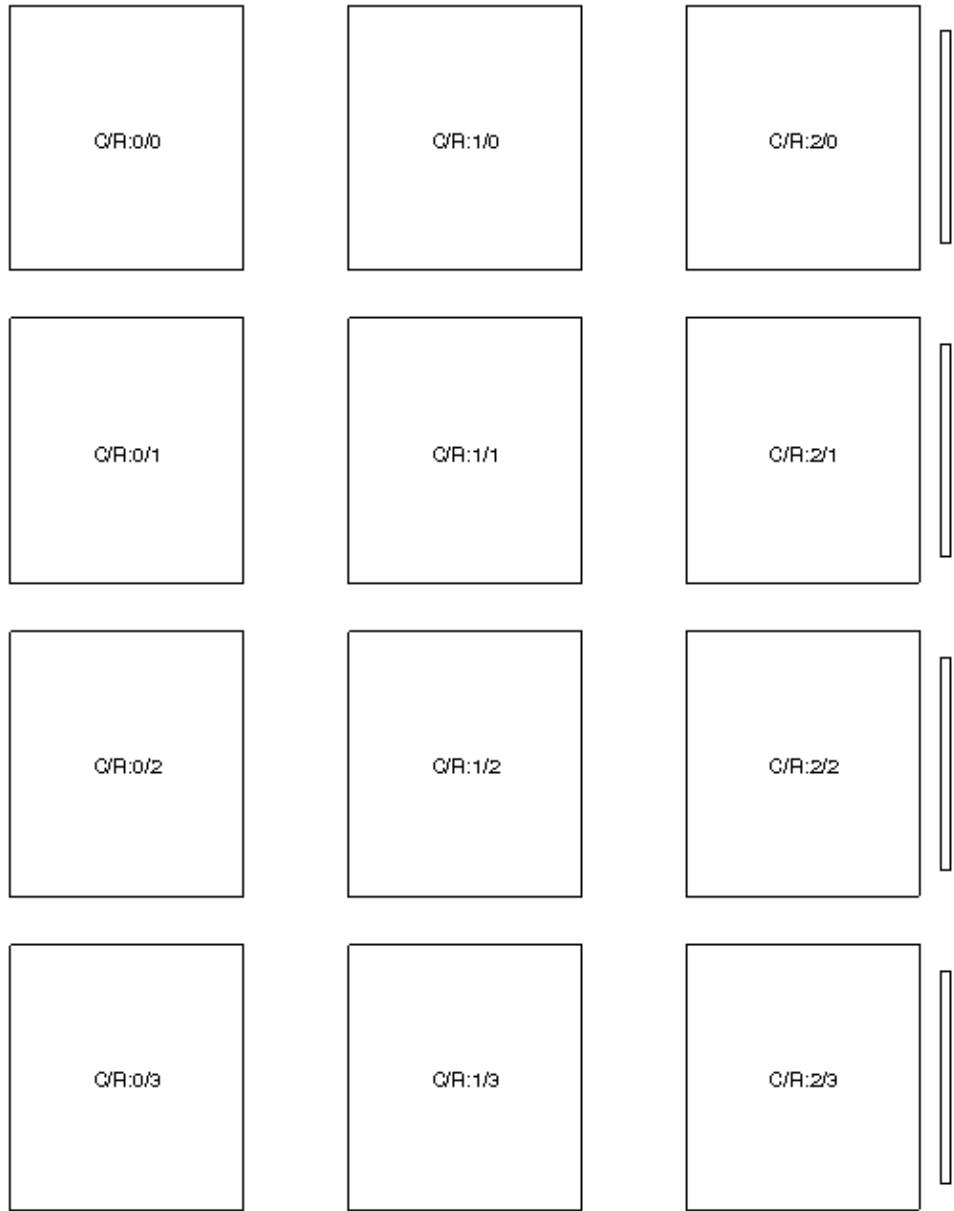
```
import cdms,EzTemplate,vcs,sys
## 12 plots 1 legend per row on the right
## Initialize VCS
x=vcs.init()

M=EzTemplate.Multi(rows=4,columns=3)
M.legend.direction='vertical'
for i in range(12):
```

```

t=M.get(legend='local')
if i%3 !=2:
    t.legend.priority=0 # Turn off legend
##    x.plot(s[i],t,iso)
M.preview('test3b')
raw_input('Done')

```



In this example we show how to control the margins and leged thickness

```

import cdms,EzTemplate,vcs,sys
## 12 plot playing with margins and legend thickness

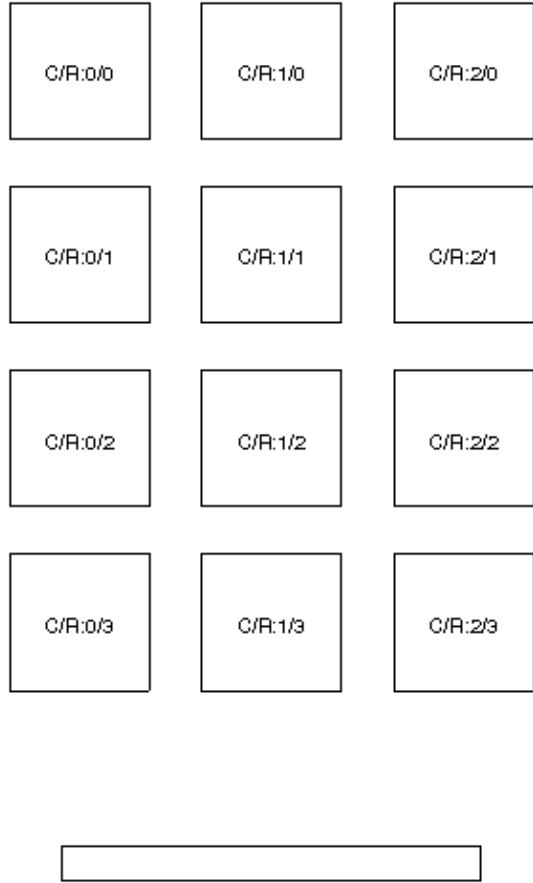
```

```
## Initialize VCS
x=vcs.init()

M=EzTemplate.Multi(rows=4,columns=3)
M.margins.top=.25
M.margins.bottom=.25
M.margins.left=.25
M.margins.right=.25

## The legend uses the bottom margin for display area
## We need to "shrink it"
M.legend.thickness=.1
for i in range(12):
    t=M.get()
    ##     x.plot(s[i],t,iso)
M.preview('test4')

raw_input('Done')
```



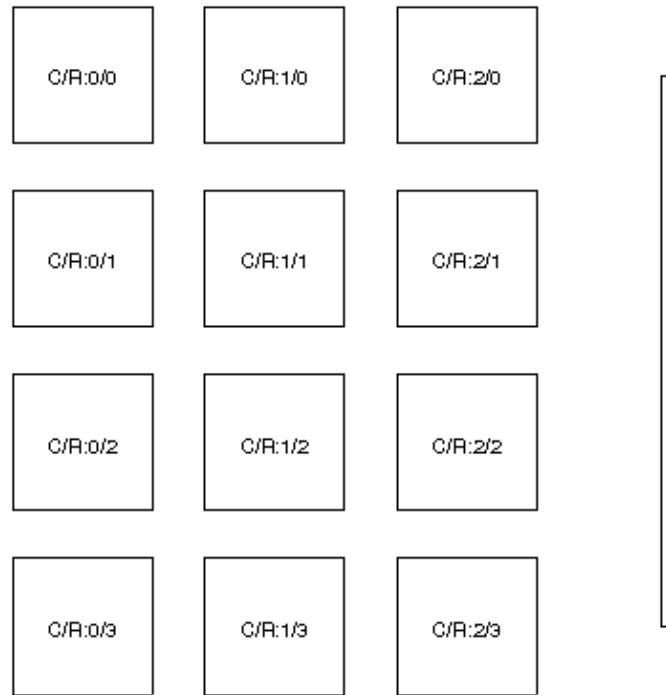
In this example we show how to control margins and legend direction

```
import EzTemplate,vcs
## Initialize VCS
x=vcs.init()

M=EzTemplate.Multi(rows=4,columns=3)
M.margins.top=.25
M.margins.bottom=.25
M.margins.left=.25
```

```
M.margins.right=.25

M.legend.direction='vertical'
## The legend uses the right margin for display are
## We need to "shrink it"
M.legend.thickness=.05
for i in range(12):
    t=M.get()
    ##      x.plot(s[i],t,iso)
M.preview('test5')
raw_input('Done')
```



In this example we show how to mix local and "global" legend and how to alternate the direction of the local legend bars

```

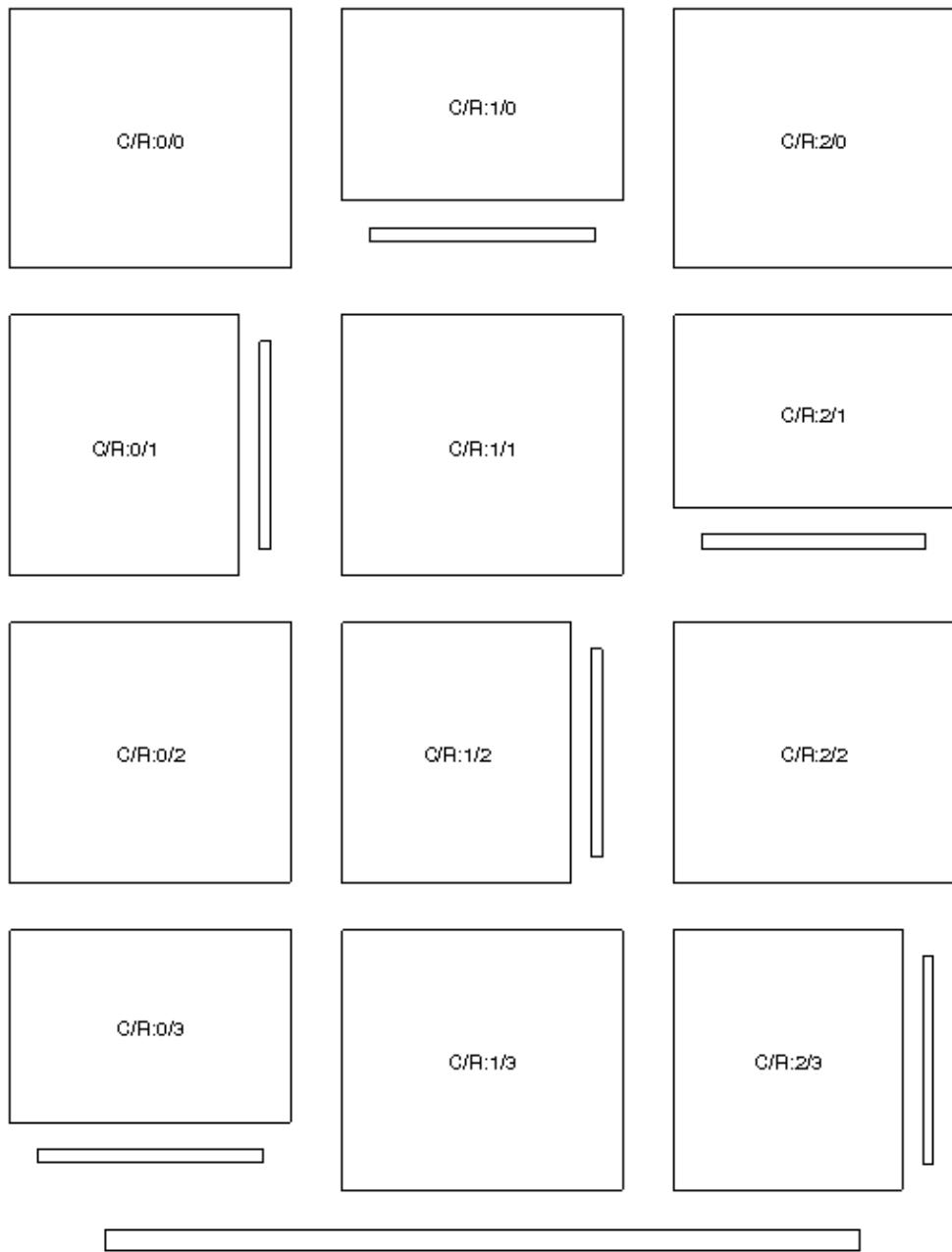
import EzTemplate,vcs

## Initialize VCS
x=vcs.init()

M=EzTemplate.Multi(rows=4,columns=3)
for i in range(12):

```

```
if i%2==1:  
    if i%4 == 3:  
        M.legend.direction='vertical'  
        t=M.get(legend='local')  
        M.legend.direction='horizontal'  
    else:  
        t=M.get()  
    #x.plot(s[i],t,iso)  
M.preview('test6')  
raw_input('Done')
```



In this example we show how to get the templates in a different order (reversed here)

```

import cdms,EzTemplate,vcs,sys

## Initialize VCS
x=vcs.init()

M=EzTemplate.Multi(rows=4,columns=3)

icol=3

```

```
irow=4
for i in range(12):
    if i % 3 == 0:
        irow-=1
    icol-=1
    t=M.get(column=icol,row=irow)
#    x.plot(s[11-i],t,iso)
    if icol==0 : icol=3
M.preview('test7')
raw_input('Done')
```



In this example we show how to control the "spacing" parameter

```
import EzTemplate,vcs

## Initialize VCS
x=vcs.init()
iso=x.createisofill('test')
iso.levels=vcs.mkscale(0.,100.)
iso.fillareacolors=vcs.getcolors(iso.levels)
```

```
M=EzTemplate.Multi(rows=4,columns=3)
M.spacing.horizontal=.25
M.spacing.vertical=.1
M.preview('test8')
raw_input('Done')
```

